Method and system for Automatic Documentation of Configurable Systems.

The present invention relates to computer systems in general, and more particularly to systems and methods generation and management of automatic documentation for configurable computerized systems.

Background of the Invention

Collaborative configurable systems are computer software systems, computer hardware systems or combinations thereof, wherein a system behavior is controlled and modifiable by a set of configuration data. Examples of such systems include, (but are not limited to), database management systems, electronic messaging and mail software and servers, electronic commerce systems, computer operating systems, (especially multi user system such as Unix® or Microsoft Windows NT®), database management systems, configurable network equipment such as routers switches, and the like. Those systems may comprise a single computer or multiple, interconnected computers, known as multi-server systems.

Configuration data is a collection of configuration parameters and data objects that control and modify at least a portion of the configurable system behavior. Complex configurable systems typically have a large number of such configuration parameters, often ranging in the hundreds and sometimes thousands of parameters. Often cross-dependencies are observed, where a change of one parameter will cause unforeseen and unintended system behavior requiring change to other parameters. Configuring such systems is difficult, and requires a high level of knowledge, skill and experience. The problem becomes exponentially more complex when several such systems are connected together, for example a number of cooperating computer servers, distributed e-mail servers, distributed database systems, and cooperating network switches and routers or any other distributed configurable system. In those cases, changing a parameter on one system may even disrupt the operation of another.

Due to the complexity involved in the configuration, once configured, the configurable system becomes a system of its own merit, a special case separate from the general case of the generic non-configured system. While the configured system behavior may be constructed and understood using documentation for the generic system

combined with configuration data, such construction is time consuming and often wasteful and expensive. Therefore it is highly desirable to document the specific case of every instance of the configured system separately, in order to ease maintenance and troubleshooting, as well as to facilitate knowledge transfer to new personnel, and generally increase operational efficiency.

Generating such documentation is an extremely tedious job that often requires a highly trained professional to perform. Since the configuration of such systems changes with time, the task is a continual one, taxing information technology personnel. And thus automatic generation of such documentation as shown by the present invention is clearly advantageous. In these specifications, the word "automatic" and its derivative means an operation that is, or may be, created or coordinated primarily by a machine or a computer, especially as compared to chores and tasks that were formerly done primarily by human labor. In an automatic documentation generation step for example, little or no manual user intervention is necessary. Manual intervention implies steps such as manual data entry of configuration parameters, repeated text entries, etc.

Constraint programming (alternatively referred to as "constraints-based reasoning", or "constraint satisfaction") is a technique, most appropriate for computer use, which is based on creating a model of a problem in terms of the requirements for a solution. By defining acceptable values to variables and constraints that define and optionally quantify the allowed relationships among the values assigned to variables, a formal representation of the problem is produced. Standard constraints programming methods can use this representation to find a solution to the problem. The technique also offers an efficient approach to problem solving by making inferences on possible solutions. The technology is well known in the art. By way of example, constraints programming was used as an aid to system configuration as described in US patent 5,708,798 to Lynch et al. and in PCT/US97/21218 to Elfe et al. Constraint programming easily lends itself to diagnostics and also to the automatic documentation process described herewith.

While most configurable systems provide means for reading and printing the system configuration data, a distinction should be drawn between raw configuration data and documentation. Raw configuration data often relates to a single system in a multi-system

environment, and is mostly a cryptic list of variable names with their associated values, with little if any explanation of their meaning. An excellent example of raw configuration data may be found in Windows Registry, which is a common configuration space for the Microsoft Windows® operating systems. Documentation on the other hand, is an organized collection of knowledge that, separately or in combination, represents the state or configuration of a system, teaches system operation, and aids in troubleshooting. Documentation is characteristically constructed of complete sentences in a human readable language, with appropriate punctuation, paragraph and optionally section separation. Typically, documentation also includes access tools such as a table of contents (TOC) or an index to further ease access to specific sections of the data. Generally, documentation also attempts to explain the meaning behind the configuration parameters, optionally including the relationships between multiple systems and the meaning of several key parameters and their effect on system operation. Another optional feature of documentation is the ability to propose configurable parameter values for sample systems, or, in the case of the current invention, the system being documented. Optionally, drawings depicting the condition and interrelationship between the described system elements further enhance the documentation. It should also be noted that documentation is characterized primarily by its content and organization, and not by the medium on which it is presented. Thus documentation may be printed on paper, viewed or edited on a computer screen by the likes of a word processor or a World Wide Web browser, etc. Whether viewed on a computer screen or printed on paper, documentation helps preserve and manage the knowledge accumulated in an organization. It thus increases operational efficiency, helps educate new members, and aids in problem solving and design for future expansion.

There are currently several aids to grouping and displaying raw configuration data relating to computer systems. One such example is available from MessageWise® in Ottawa, Ontario, Canada. This system reads configuration data from several Microsoft Exchange servers, and places them in a database. Queries can then be placed against the data to find answers to specific question. However the MessageWise system does not provide a well-documented meaning of a textual and visual description, organized in an easily read document as described above.

Another configuration data collection tool named Emap and distributed by Microsoft® Corporation of Redmond WA, provides a graphical representation of connections between exchange servers in an organization. Similarly, several products allow representing an organizational data or voice network by drawings representing the network structure. These types of products are exemplified in US patent 5,926,463 to Ahern et al.

None of the above solutions provide documentation as described earlier in these specifications. All lack the effects and efficiency of teachings that a well organized textual and visual document provides, and thus do not provide the dept of knowledge preservation and knowledge management provided by such documents. It is clear therefore that there is an unfulfilled need in the industry to provide automated documentation generation, re-generation, and management tools for highly configurable systems.

Summary of the Invention

It is therefore a goal of the current invention to provide a method and a system for automatic generation of documentation for computerized configurable systems.

The embodiments of the invention derive documentation from a combination of generic system knowledge and the configuration data of a specific system instance, such as one or more configured systems. They thus retrieve or gather, configuration parameters into a computer, and the computer selects predetermined explanatory text segments corresponding with relevant configuration parameters. (Retrieve, collect or gather are used interchangeably in this application and in this context of gathering configuration parameters from a configurable system). The computer outputs the selected text segments together with the corresponding parameters, thus forming a document specific to the configurable system. Preferably, the output is divided into sections and paragraphs where the parameter or a group of parameters, associated with a specific segment or paragraph is outputted in proximity to each other. Furthermore, the current invention preferably groups similar interrelated parameters and their associated text segments in proximity to each other.

Text selection is generally done by a template, implemented by any convenient method. The template will generally include placeholders for the configuration variables and their values, and optionally drawings and other explanatory material. The template may be implemented as a collection of text stored in a file, embedded within the program code, or in a database. Alternatively or in combination, the template may comprise text segments included as print statements that are selectively executed according to specific system configurations and other environmental conditions such as system location, ownership data and the like. These specifications relate interchangeably to different methods of producing a document from predetermined text segments merged with configuration parameters, as utilizing a template. Specific embodiments may use different methods as a matter of technical choice.

Preferably, the invention also automatically generates tools to ease the access to the information in such a document. Preferably, the invention provides a table of contents (TOC) detailing the relative locations of certain sections of the document. More preferably, an index of selected parameters and sections in the document is also provided, detailing their relative locations. Both the table of contents and the index may be adapted best to the document media, for example a printed TOC and index when the document is printed on paper, and alternatively hyperlinks or other computerized access methods may be used when the documentation is viewed on computer screen.

The invention is easily adaptable for various types of configurable systems having configurable parameters accessible by computer, especially for configurable software applications. Listed below are but a small sample of the systems and system types for which the invention is especially adaptable and beneficial: electronic messaging and mail delivery systems such as Microsoft Exchange® (Microsoft Corp, Redmond, WA), computer operating systems such as Microsoft Windows NT® or Windows 2000® and different varieties of the Unix® operating systems, groups of computers such as one or more Windows NT domains. Similarly, complex database management systems such as DB2® (IBM Corp, Armonk NY), Informix® (Informix, Menlo Park, CA), or Microsoft SQL Server®, and groupware applications like Lotus Notes® (Lotus Corp, Cambridge, MA) and GroupWise® (Novell Corp., Provo, Utah) all lend themselves to being documented by the present invention. Additional examples of configurable systems that

may benefit from the current invention include electronic commerce applications, enterprise management systems such as SAP® (SAP AG, Waldorf, Germany), BAAN (BAAN Corp, Barneveld, Netherland) and Peoplesoft® (Peoplesoft, Pleasanton, CA) and enterprise storage systems such as EMC²® (EMC Corp, Hopkinton, MA) and Legato® (Legato Corp, Palo Alto, CA), and network nodes such as Cisco® routers (Cisco Corp, San Jose, CA).

Preferably, embodiments of the invention generate documentation by retrieving the configuration data of one or more configurable systems and providing a computer readable set of rules describing knowledge about the system. The rules has embedded in them a set of acceptable values associated with specific configuration parameter(s). The invention then compare the values of configuration parameters against the values expressed in the associated rules, and output an error or warning condition if a parameter violates those rules. The acceptable values may be fixed in the rule, or alternatively, may be computably modified based on portions of the configuration parameters. In addition to acceptable values, the program may also include indications of desired values that may be outputted as needed.

A method for providing the set of rules and the knowledge functionality as described above may be provided by a computer readable generic data model of the configurable system. The generic model corresponds to the interrelationship of the sub-components of the configurable system and thus reflects its underlying structure. After the configuration data is retrieved from the configured system, an instance model is constructed in accordance with the generic model. Optionally, the generic model incorporates the system knowledge described above, and includes sets of acceptable values and other interrelationships between the system sub-components. Preferably the system knowledge is implemented utilizing constraints programming logic and techniques, which allows for fast and efficient checking of a complex set of requirements, and easy flagging of error conditions. The generic model also facilitates the creation of a document that is organized in a manner most closely related to the structure of the configurable system, and further enhances the readability and reduces the time required to comprehend the system.

In a preferred embodiment of the invention, the raw data is parsed by a data parser program module, entered into a database, and merged into a unified or segmented text template. Rules are then coded into the program to specify values, method of computing values or locations of finding the appropriate values, against which the configuration parameters of the sampled system may be checked. This method allows the user to change the presentation of the document by embedding the results of queries against the database into the document in ways the user best believes will enhance system understanding or troubleshooting. Using such a database is advantageous due to the simplicity and low cost of the implementation.

An alternative object of the current invention is to provide identification of any configuration mismatches or potential problems in a configuration and to flag those conditions to the user. Optionally, optimization suggestions may be made based on the quantification of different aspects of the configuration data and the generic system knowledge.

Another alternative object of the invention is to provide means of comparing current and previous configurations and to point out differences between them. Such comparisons are useful during the troubleshooting of a system, which malfunctions due to unknown reason. By comparing an old, "known good" configuration with the current configuration, differences become apparent and, due to the documentation generating nature of the current invention, such changes may easily be placed in context and thus aid in determining if any of the modified parameters may have caused the malfunction.

It should be pointed out specifically that the invention, with its several capabilities, applies to multi-server configurable systems as well as single server configurable systems. This is a clear advantage since, as mentioned earlier, the system complexity grows exponentially with the number of servers. Examples of such multi-server environments are a plurality of server computers operating under the Windows NT operating system within a single domain, or multiple communicating domains with or without trust relationships. Another example is one or more Microsoft Exchange electronic mail servers in an organization. Similar in nature is a set of co-operating

configurable network nodes such as routers, switches, concentrators, frame relay access devices, and other nodes comprising a data or voice network.

It should also be noted that the invention may be integrated within the configurable system itself, either as an integral part of the system software or by having a separate program active on a computer that is a part of the configurable system.

The system operates generally by collecting configuration data from individual systems or from system sub-components that comprise the configurable system to be documented. Preferably, a collector program is deployed to collect the configuration data. A collector program is a program constructed to collect configuration data from one or more configurable systems, and is constructed specifically for each platform or configurable system and collects the raw data for transmission into a Documentor. The configuration data is then preferably encrypted (especially in the case of transmitting the data over the Internet) before being transmitted to the Documentor.

A Documentor, (alternatively referred herein as a Documentation Server) is a computer program that receives configuration data as its input and generates documentation as its output, primarily when receiving the data from a configurable system and generating documentation pertaining to the system. It performs this function by outputting a template, or pre-determined text segments relating to configuration parameters or to a group of inter-related configuration parameters. The text segments of which the document is comprised are primarily explanatory in nature and are composed to improve the understanding of meaning and effects of the configuration parameters to which they relate. Optionally, drawings and tables are also outputted as needed to increase the understanding and readability of the produced document. The drawings, tables, and text segments are outputted to paper, a computer screen, the Internet, or any other appropriate output device or method. Additionally the Documentor may output the document to a computer file.

In one alternative embodiment for a Documentor, a generic operational model is provided. The operational model embeds therein information and knowledge which is required for a proper generic system operation, and may be viewed as a set of generic implementations of operational configurations specific to the generic system. The

generic model therefore, details the underlying structure of the interrelationships between different sub-components of the generic case of the configurable system. Each allowable configured system is a subset of the generic configurable system. Similarly, each model of a configured system constructed according to the generic model is a subset of the generic model that reflects the configured system underlying structure as configured. Such specific models of configured systems are called "instance models".

Currently it is believed that such model is best created utilizing constraints programming, wherein the knowledge is expressed as a set of constraints on the system's configuration parameters. However as will be seen below, other, less complex methodology may be applied, both to model creation and to the primary goal of automatic documentation generation, without detracting from the invention. In a constraints-based implementation, the collection of configuration data is examined in light of the constraints expressed in the generic model and is entered into an instance model. Once completed, the instance model is a representation of the specific instance of the configurable system as configured.

Parameters that violate constraints are flagged as an error, and may or may not be entered into the instance model. Optionally, parameters may be assigned values according to the level into which they fit the generic model, and such values may be utilized later to propose changes to optimize the configurable system.

In one preferred embodiment, the collector program is downloaded onto a computer connected to the configurable system. The program then collects configuration data from all relevant individual systems comprising the configurable system. The data is transferred via the Internet to the Documentor that processes the configuration data and prepares the documentation. The documentation is then made available to the user over the Internet. Optionally, the collector program may be operated from within a World Wide Web browser and starts operating automatically after being downloaded.

In a most preferred implementation, a documentation server comprising the collector program and the Documentor portion is coupled to a local area network or an Intranet that is in communication with the configurable system to be documented. An Intranet is one or more interconnected computer networks that are characterized by being primarily

within the electronic bounds of an organization. Typically electronic access to an Intranet is not open to the general public, and it thus affords a protected computing environment for storing and accessing organization confidential data. Many Intranets utilize Internet like technologies, and some utilize communications provided by public communication means, for example, a dedicated line in a PSTN network. However data access to an Intranet is controlled from within the owning organization.

Coupling a documentation server directly to an *Intranet* overcomes a disadvantage of an *Internet* coupled server by maintaining the configuration data within the organization-controlled network. In an Internet coupled server the data, even if encoded and encrypted, may present a security risk for certain configurable systems. An Intranet coupled documentation server, especially when implemented as a dedicated computer, becomes a *Documentation Appliance*, operating within the confines of an organization. The Documentation Appliance is readily available to company personnel, offers local system documentation and storage of previous configurations, all while avoiding the need to send sensitive data over public networks.

Such a Documentation Appliance is preferably a dedicated server that may be adapted for documenting one or more types of configurable systems. It is also preferable to have such a Documentation Appliance keep logs of document production and that such logs will be remotely accessible to ease debugging and service. The Appliance may be coupled directly to the configurable system or may utilize an intermediary computer.

A Documentation Appliance server attached to an organization Intranet may also be configured to retrieve the configurable system configuration information according to a pre-determined schedule, and activate updated documentation production automatically. Together with the capability to compare previous and current sets of configuration parameters, a Documentation Appliance may thus produce a list of changes that occurred between activations, in effect to create a log, tracking modifications and changes in the configurable system.

There are clear advantages to accessing the documentation via a computer. To facilitate that, the invention also provides for a World Wide Web type server that is constructed to distribute the documents produced by the current invention to a web

browser. Since the documentation in the preferred implementation is produced by a print engine capable of producing HTML, the integration of a web server within the appliance allows any person with access rights to browse, search and view the documentation as needed.

In yet another embodiment of the present invention the program code required for performing the steps required for creating the document may be integrated into the configurable system itself. In such an embodiment the configuration data is already available to the document generation module. The document generation module comprises primarily of a text template, implemented as program code, text segments, or any other convenient manner, and program code to merge the configuration data within such template and output the template with merge data as a document. The program code may be integrated into the configurable system as an integral part of the configurable system code or may be implemented as an add-on module loaded and executed by the configurable system. Other embodiment details may also be integrated within the configurable system as described below.

Brief description of the drawings

This invention may be better understood from the descriptions and claims which follow, and from the accompanying drawings in which:

Fig. 1 depicts a general schematic component and data flow diagram of a preferred embodiment of the invention

Fig. 2 is a generalized flow diagram of the collector program portion of an embodiment of the invention.

Fig. 3 is a flow diagram for generating an instance model in accordance with the invention

Fig. 4 depicts a generic model object in accordance with an embodiment of the present invention.

Fig. 5 is a partial and simplified 'C++'-like code of several classes used in model creation in one preferred embodiment.

Fig. 6 depicts 'C++'-like code of two 'document' methods referred to in Fig. 5 classes.

Fig. 7 depicts an example of a simplified portion of text template for another preferred embodiment of the present invention.

Fig. 8 is partial pseudo code showing a method for generating warning messages to the user according to the present invention.

Fig. 9 is a screen depiction of web based screen from which the documentation operation begins.

Fig. 10 is a screen depiction of an HTML page prior to sending collected data to documentation browser.

Fig. 11 depicts a partial sample of a table of contents for a document generated in accordance with the present invention.

Fig. 12 depicts a partial sample of documentation generated in accordance with the present invention.

Fig. 13 depicts another partial sample of documentation generated in accordance with the present invention.

Fig. 14 depicts another schematic implementation of the invention utilizing a database and a text template in accordance with the present invention.

Fig. 15 depicts a simplified diagram showing connection method of a documentation appliance in accordance with the present invention.

Detailed description of the invention

This invention will be explained with reference primarily to Microsoft Exchange® and Windows NT® platforms by way of example only. Microsoft Exchange is a popular electronic mail server (hereinafter "Exchange"), and Microsoft Windows NT is a popular operating system (hereinafter "NT" or "Windows NT" interchangeably). It will be clear however to those skilled in the art that primarily the platform specific data need to be modified in order to modify the documentor portion of the invention to handle other types of configurable systems. Such platform specific data includes appropriate text sections

and appropriate constraints or rules to define the underlying structure and the acceptable variable values and relationships. Similarly, it will be clear that while the collector portion of the current invention is platform and application specific, adapting the collector program described in this application to operate in a specific platform is a matter of common skill.

In its most basic form, the invention involves a database containing a set of predetermined text sentences or paragraphs, associated with at least one specific variable. A collector program is provided for reading configuration parameters from a configurable system, and the associated text paragraph is outputted together with the configuration parameters. Optionally entries are made in a table of contents for text paragraphs, and the table of contents is outputted as well. The output may be to a printer, a screen, or a file. Output may also utilize the World Wide Web technology to offer organized access to the documentation, in which case the table of contents is best composed of hyperlinks to appropriate sections. Another alternative output forms include audio and video output. An audio output may be generated by a text to speech translation module. Video output require a video template, similar to the text template described herein, but adapted for video signals and for embedding external values such as configuration parameter values, in a predetermined video stream.

As noted above, the invention may also be practiced by integrating the steps described herein into the configurable system code. For example, a module may be integrated into a Microsoft Exchange server that will retrieve all configuration parameters directly from the Exchange system within which it operates. Alternatively, a Documentor program may be operated on a computer that is a part of the configurable system. For example, a Documentor program may be operated on a computer utilized as a Windows NT domain controller, where much of the configuration data is locally available in a system adapted for documenting a Windows NT environment. The document generation steps described herein are similar with the primary difference being the location where certain portions of the invention are executed.

While novel and useful even at this basic level, the invention further benefits the user by grouping similar features and/or variables together, for example listing all domain

controllers in an NT domain, or creating a list of all mail transfer agents (MTA) scheduled to run in an Exchange organization, including their prospective protocols or schedule. Optionally, the Documentor portion may collect similar information from multiple servers. So, for example, replication information between database servers may be listed together to ease understanding of the proper propagation order. Such information will preferably be grouped in close proximity with several paragraphs explaining the process of propagation and the meaning of each of the related variables. As further explained below, utilizing constraints or rules eases verification of proper configuration.

The preferred implementation will contain general data explaining the generic operation of the configurable system, together with system specific data as configured. Preferably, generic and specific advice relating to configuration will be placed in proximity to system information relating specifically to the system being documented.

For an explanation of the operation of the invention, reference is now made to the drawings, depicting preferred embodiments of the invention, shown here for illustrative purposes only, primarily for operation on a Microsoft Exchange platform.

In Fig. 1 the documentation server 100 in a preferred implementation is located remotely to the client Exchange organization 170. The documentation server 100 is connected to Exchange organization 170 via the Internet or an Intranet. A client workstation 150 is the intermediary between the documentation server 100 and the Exchange organization 170. It will be clear however, that the client workstation 150 may be an integral part of the Exchange organization 170 or even an Exchange server per se, such as 174, 180 or 185. The client workstation 150 enables access to the Exchange servers with sufficient access rights to allow a program running thereon to access the required configuration data in the registry and in the Exchange directory. Additionally, in this preferred implementation, the client workstation should have an operational web browser 160 from which documentation production operations may be initiated.

The exchange organization depicted in Fig.1 is a generic organization having at least one Exchange Source Server 174, and any number of servers, only two of which are

shown, designated as 180 and 185. The Exchange directory 176 resides on the source server 174.

Utilizing the process begins with a client creating a connection from client workstation 150 to the documentation server 100 via the web browser 160. The client than downloads 11 a copy 125 of the collector program 120 to the client workstation. In this preferred embodiment the collector program is constructed as an ActiveX ® program or a Java® (Sun Microsystems, Palo Alto, CA) program, and the copy of the collector 125 begins operating automatically after it was downloaded.

Fig. 2 depicts the operation of the collector program. After program initialization 200, the collector program 125 connects to the Exchange Source Server 174 as depicted in step 12 in Fig.1, and extracts 220 the Exchange Directory information 176. Next, the collector program proceeds to extract 240 registry data 178 from the Exchange Source Server 174 in step 13, and retrieves registry data 182 and 187 for all other Exchange servers in the organization, as shown in steps 13 a, b ...n in Fig. 1. Clearly, while Fig.1 shows only three servers, this depiction is for explanation purposes, and the number of servers may vary from one to any number supportable by the configurable platform. For increased efficiency, the steps of collecting the configuration data from the various servers in steps 13b through 13n are carried out simultaneously.

Collector program 125 then creates 250 an HTML page 255, compresses the data collected in steps 12 and 13 a...n, and encodes 260 the data for compatibility with transmission using the HTTP protocol. The data is than embedded in HTML page 255, preferably as hidden text. The collector program 125 than loads HTML page 255 into web browser 160, and points 270 client web browser 160 to the page 255. An example of the HTML page 255 when acted upon by the web browser as depicted in Fig. 10. The user is given the option, by activating button 1010, to send the data to documentation server 100.

While the mechanism of collecting the configuration data shown in steps 220 and 240 is described uniquely as applicable to Exchange environment, those steps constitute the mechanics of collecting configuration information specific to the platform. The function of automatic collection of the configuration data may be performed by any known means

without departing from the invention. Adaptation of the collector program described above for different platforms requires only regular programming skills and general knowledge of the platform. Alternatively, the collector program may be constructed to operate through other communication and collecting methods such as SNMP for network nodes, in which case no HTML communications is required and is often unavailable. In the case of the Documentation Appliance, the collector program may be implemented as a collector program module operating on the Appliance, with the configuration data retrieved directly from the configurable system by the module.

Fig. 3 details the operation of instance model creator 134. The model creation happens within the document generator 130. Once the user sends the data embedded in HTML page 255 to the documentation server 100, the program retrieves 310 the information from the web page, decodes and decompresses 320 the data. The data is then parsed into variable names and corresponding values in steps 330 and 350. The raw data is stored in a database 132. Doing so allows easy approach to different data views as needed. The parsed data is then presented, one variable at a time, to step 360 in which an instance model is created.

In a preferred embodiment the instance model 145 is an object-oriented artifact assembled from instances of predefined object types described in a generic model 140 that is incorporated into the documentation server 100.

The generic model 140 for a specific application domain (e.g., Microsoft Exchange, SAP database, or a specific computer operating system) consists of a set of classes describing objects available in the configurable system universe. Thus, generic models are specific to a particular configurable system domain: for example, at least one generic model is provided for Microsoft Exchange, while another one is provided for Windows NT, and so on.

Each type of object in the configurable system universe that is relevant to the task performed by the current invention is represented in the generic model in the form of a class. The class describes its composition (in terms of lower level objects) and structure (in terms of relationships among its components, e.g. Part-Of, Is-A, etc.). Optionally,

behavior can also be expressed as part of the model, in the form of additional relations among constituent parts and/or attributes, describing correct and/or faulty functioning.

By way of example, in a Windows NT domain, there will be a library object describing a server, and certain server characteristic parameters such as name, version information, and the like. Constraints may be attached to each of those variables, for example a constraint that the domain name has to be the same for the server as for the whole domain is needed to verify that the domain is consistent. However more complex requirements, for example communications capability between servers, require that a whole class of objects dealing with network communications be enabled and consistent. Such structural demands comprise the structural generic model 145, together with a set of constraints that are applied to this underlying structure. By adopting the generic model and the object library to specific platforms, such as mail servers, operating systems, network equipment, etc., the documentation server 100 can be modified to operate on various platforms.

A partial example of several objects relating to a model for the Microsoft Exchange platform is shown in Fig. 5 in 'C++'-like notation. By way of example, class 'Organization' 510 is the root of the model, and comprises internal attributes such as the organization name 511, and other subservient objects comprising the rest of the organization model. In this example, the vector of subservient objects is of type 'Site' shown as 516. The class Organization 510 also incorporates several methods. The posting step described below is carried out in the method 'post' 512. Rules or constraints applied to the object to check the validity of the object against the generic model are applied using the 'validate' method 513, and the method 'document' 515 is used to create the documentation that is specific to the configurable system instance.

Examining class Site 520 shows a close similarity to class Organization 510 above, with the major structural differences being the members included in the class. While the Organization class 510 included a vector 516 detailing a list of sites, each Site object details a list of Server objects 526 and a list of Protocols objects 524 with similar collection of methods. In class Server 530 we see a similar pattern yet again. Class Protocol 540 also incorporates similar methods, but being a leaf of the hierarchical model

structure, it does not include any vector of inherent objects connected thereto. It will therefore be clear to any person familiar with the organization and operation of the Microsoft Exchange software, that the construction shown in Fig. 5 represents not only a collection of objects of the types described above, but also the underlying structure of a Microsoft Exchange installation. It will also be clear that similar constructs may be created for other configurable systems such as a Windows NT domain, a network organization, Lotus Notes software, database servers etc.

In the process of instance model creation 360, each applicable configuration variable is examined against the generic model. In this step, known as the 'post' or 'posting' step, an object instance is created for each relevant configuration variable. Variables that conform to the generic model and pass the constraints imposed on the instance object are posted in the instance model. Posting occurs in an object-oriented manner that maintains a structural semblance to the underlying principles defined in the generic model for the platform as described above.

Constraints may be divided primarily into two categories: hard and soft constraints. Hard constraints are such that a violation thereof may prevent the system from operating, while soft constraints are mainly related to performance issues, and a violation of such soft constraints will allow the continued system operation, albeit in less than optimal manner. Whether a constraint is a hard or soft, is particular to each constraint and is responsive to the value of one or more variables. Additionally, as is well known in the art, constraints may be quantitative, and assign values corresponding to a level of conformance with the generic model. For example, such values are utilized for providing the user with suggestions and warnings related to optimizing the configurable system performance and/or to modifications required for maintaining optimal system operational status.

Fig. 4 presents a simplified view of the document construction carried out in the document generation module 138 portion of the documentation generator 130. A documentation template 430 describing and explaining different aspects of the system is created and stored in the documentation server 100. The template may be stored in separate files, as part of the model, or as data available to the documentation generator

138 in any convenient format. Internal attributes and sub-components are integrated into the template as needed. Figs. 12 and 13 are sample documentation pages, while Fig. 11 is a sample of a table of contents; all generated by the current invention. As can be observed in Figs. 12 and 13 drawings may also be embedded in the template.

Fig. 6 is a partial listing of a computer program, written in a 'C++'-like language. The numeral 610 designates a simplified sample of the 'document' method 525 code of an object of class 'Site' 520. The method uses the 'C++' standard output stream 'cout'. Portions of the documentation template 430 are embedded in the first output statement 613 in the form of text strings like "The site" 618. The output statement 613 also includes internal attributes 420 such as the variable 'm_name' 615 and sub-components 410 such as the method 'm_servers.size()' 617. Additionally, the method calls other documenting methods in all servers in its 'm_servers' lists seen in 614 and all protocols in its 'm_protocols' list as seen in 616. While simple attributes such as 'm_name' are outputted directly from class members or global variables, other outputs occur from methods internal as well as external to the method. Since each 'document' method dictates the order in which the subservient objects print, a very orderly and well-organized document can easily be produced. Attention is also called for the 'endl' statement 619 at the end of the cout statement 613, which is an example of the formatting information imparted by the program. More complex formatting, merging information, tables and drawings may be incorporated from such predetermined data or taken from files external to the program to be merged in the output document.

Fig 6. Also shows a method of generating the table of contents or TOC. Statement 611 supplies the current location in the output stream to a procedure named TOC_Entry, which stores the context and associated location for further generation of the table of contents. Similarly, the procedure "Index_Entry()" provides the index context, in the example the site name and the location within the output document. If for example the standard output stream 'cout' statements is redirected to one computer file, both the TOC and index generating procedures simply output the indexing information into one or more separate index files. Generating an index and TOC from such files is a simple procedure, and the TOC and index may be created according to the viewing method, e.g. by

hyperlinks that are dynamically generated when the document is read on screen, or by printing the information when the document is reproduced on paper.

Fig. 6 also shows another example of the document method common to most if not all nodes in the model. In the 'Server::document' method 650 output statement 660 generates documentation output without calling any external method.

In a preferred implementation, the output produced by the Documentor is a set of printing instructions for a virtual printing machine. Using specific drivers, the documentor is then able to produce documentation in different formats such as HTML, Postscript, PCL, Latex®, Adobe Acrobat® Microsoft Word® document, or any other desired format. The language for the virtual printing machine contains, among other constructs, commands for describing the document layout, such as chapter, section, subsection, and similar constructs. Those commands automatically create the appropriate entry in the optional TOC and index. Other commands for the virtual printing machine include commands for specifying document composition, such as structured elements such as tables and bulleted lists, and/or unstructured elements such as paragraph, note, warning, and hints. Additional virtual printing machine commands include formatting instructions like vertical/horizontal space, indentation justification and the like, as well as font management functions such as color, bold, italic, etc.

The virtual printing machine is implemented using object oriented programming techniques, which provides a multiplexer class. Implementing the commands described above in a manner similar to the standard 'C++' stream manipulators and overloading the '<<' operator, from the point of view of the Documentor, the multiplexer class is used the same way as standard 'C++' streams. The virtual printing machine also provides several driver classes, one for each type of format or formatting language. Each class implements the translation of the printing commands to the target formatting language.

A simplified description of the output process is as follows: The Documentor creates one multiplexer instance to which it attaches instances of drivers for all the required types of output. Every time the documentor uses the output '<<' operator, the desired information or printing construct is sent to the multiplexer which distributes it to all the drivers currently attached to it. Each driver than creates an output according to its specific

formatting language. The drivers also create the table of contents and index in a desired manner that fits the formatted output, such as HTML style hyperlinks for an HTML document or as formatted and printable TOC and index for paper output.

Optionally, the invention may further use the raw data database 132 to allow user specified view of data specific to the configurable system in question. The raw data database may best be managed by a database management system against which queries may be posted, and in which the data retrieved forms the basis for the view. The creation of such a view will allow generating partial documentation, for example detailing the entire information specific to a single subnet in a network. This allows users to concentrate on limited areas of interest; for example, a network under direct supervision of a specific user. This feature may be further enhanced by the addition of pointers from the database to related objects in the instance model, since only the relevant portion of the instance model can easily be created as separate documentation with grouping of similar parameters in stored separate subsections of the model. Query results may also be embedded in the output document either according as predetermined by the template or according to user preferences, in a system that includes a user interface portion to accept user input for selecting document generation options.

In a simpler embodiment of the present invention depicted generally in Fig. 14, the process of document generation does not include constraint-based model creation. In such an embodiment the raw data is placed in raw data database 132 and is associated with parts of the documentation, (or text segments) that form template 1450 that correspond to the variables relevant to the documentation of the configurable system. Alternatively, template 1450 may be a predetermined document template, which includes placeholders for the results of specific queries such as the names of all servers, name of all domain controllers, etc. A documentation generation module 1410 reads the template, and when a query is encountered, it is parsed, presented to the database 132 and the results outputted to the document by a documentation generation module 1410. Fig. 7 presents a simple portion of a template to generate partial documentation for a Windows NT domain. Queries to the database 710, 720 and 730 are enclosed between character strings '<<<' and '>>>'. When the documentation generation module 1410 encounters a query such as 710, it parses the query and the results are inserted into the output stream.

The documentation generation module 1410 can also parse constructs such as variable name resolution and substitution, and an if statement as shown in 720, and decide which of the two statements 730 or 740 to output. The example depicted in Fig. 7 shows a simple but effective method to automatically generate elaborate documentation from the collected configuration data without the use of a complete object oriented model.

Optionally, a simple error checking and reporting routine may also be incorporated, in which a collection of rules is stored in the program and relevant variables are checked against those rules. Fig. 8 depicts a pseudo-code portion of such rule-checking program. In accordance with Fig. 8 an example 810 of such rules is to warn the user 820 if the number of mailboxes on the server multiplied by the desired mailbox disk space allocation, (or 5 MB in this example) is smaller than the available disk space. The rule may further be constructed to issue warning or error messages, or to propose more fitting values for a parameter based on the value of one or more parameters. For example, the rules or constraints may compute and output a suggestion to modify the size of a virtual memory disk swap file in response to actual RAM memory and available disk space in a server operating under Windows NT. Such proposed values may be based on an arbitrarily complex set of computations emanating from the configuration parameters or based on general system knowledge and experience.

Another objective of the current invention was previously described as comparing current with previous configurations of the configurable system to aid in troubleshooting. This objective is easily achieved by storing an older model or raw configuration data of an operational system. When a problem is observed in the configurable system operation, a present model or raw data set is collected and the new and old set of configuration data or model may be compared to each other. Outputting variables with different values between the two configurations will allow consideration of only those variables that have been modified, and thus will ease troubleshooting of the system as presently configured.

Fig. 15 depicts the Documentation Appliance described above. While the methods of document generation are similar to the methods already described, the Documentation Appliance 1500 utilizes a collector module 1520 to perform the configuration data

retrieval directly from the configurable system. The documentation appliance is also connected to an Intranet 1510 that comprises one or more interconnected networks and that is coupled to the configurable system as depicted by the Exchange Organization 170. Optionally, the documentation appliance also incorporates a web server 1560 to distribute the documentation to other computers. Optionally a scheduler 1550 is provided to activate the documentation generation according to a predetermined schedule.

Once the documentation is generated with a computer readable main body and a table of contents, additional capabilities, such as searching and editing, may be easily added to the features of the current invention as is well known in the art.

While there have been described what are at present considered to be the preferred embodiments of this invention, it will be obvious to those skilled in the art that various other embodiments, changes, and modifications may be made therein without departing from the spirit or scope of this invention and that it is, therefore, aimed to cover all such changes and modifications as fall within the true spirit and scope of the invention, for which letters patent is applied.